# Real-Time Sign Language Translation System Using Computer Vision Techniques

Pranav Balu
pxb210046@utdallas.edu

Farman Ali
fna200001@utdallas.edu

Wei Yuan Liew
wxl220016@utdallas.edu

Danny Amezquita
dxa210012@utdallas.edu

## Abstract

*Sign language serves as the primary means of communication for more than 70 million deaf and hard-of-hearing individuals worldwide. Despite this significant user base, the persistent communication barrier between sign language users and non-signers creates substantial challenges in healthcare, education, and daily interactions. This paper presents an automated system that translates ASL gestures into text in real-time, leveraging recent advances in computer vision and PyTorch-based deep learning architectures. Our goal is to focus on both accuracy and speed while maintaining accessibility through standard hardware requirements. Thus, we explored with methods that sacrificed hand tracking to further optimize speed and size of the system, but fell short as some gestures are motion-based. Despite limitations, our work contributes to the understanding of sign language translation and suggests directions for further investigation.*

## 1. Introduction

Sign language serves as the primary means of communication for more than 70 million deaf and hard-of-hearing individuals worldwide. Within the United States alone, American Sign Language (ASL) is used by approximately 500,000 people as their primary means of communication [3]. Despite this significant user base, the persistent communication barrier between sign language users and non-signers creates substantial challenges in healthcare, education, and daily interactions. This paper presents an automated system that translates ASL gestures into text and audio in real-time, leveraging recent advances in computer vision and PyTorch-based deep learning architectures.

### 1.1. Problem Statement

Despite technological advances in machine learning and computer vision, real-time sign language translation remains challenging due to several fundamental complexities:

- **Dynamic Gesture Complexity:** Sign language incorporates not just hand shapes but also movement patterns, speed variations, and spatial relationships, making it significantly more complex than static gesture recognition
- **User Variability:** Natural variations in signing styles, hand sizes, and execution speeds among different users create challenges for consistent recognition
- **Environmental Challenges:** Real-world deployment requires robust performance across varying lighting conditions, backgrounds, and camera angles
- **Real-time Processing Demands:** Effective communication requires near-instantaneous translation with minimal latency, necessitating efficient processing pipelines
- **Dataset Limitations:** While datasets exist for static ASL letters, comprehensive datasets for dynamic gestures and continuous signing remain limited

## 2. Related Work

Recent advancements in computer vision and deep learning have significantly contributed to sign language recognition. Previous efforts can be categorized into traditional computer vision techniques and modern deep learning-based approaches.

### 2.1. Traditional Approaches

Early methods in sign language recognition heavily relied on hand-crafted features and statistical models. For instance, [6] employed feature extraction techniques such as edge detection and contour analysis to classify static gestures, achieving recognition rates of 75-80% under controlled conditions. While effective for static gestures, these methods struggled to handle dynamic gestures, user variability, and environmental challenges.

### 2.2. Deep Learning Advancements

The emergence of deep learning has revolutionized gesture recognition. Convolutional Neural Networks (CNNs) have shown remarkable performance in static gesture classification. Senanayaka et al. [5] utilized CNNs for ASL alphabet recognition, achieving a high accuracy of 94% on single-letter tasks. However, their system was constrained to controlled environments, limiting real-world applicability.

To address dynamic gestures, sequence-based models like Recurrent Neural Networks (RNNs) have been explored. [4] proposed a model combining RNNs with CNN feature extractors for continuous gesture recognition, improving temporal context understanding. However, these systems often suffered from latency issues, limiting their real-time application.

### 2.3. Current Challenges

Despite these advancements, several challenges remain unresolved:

- **Dynamic Gesture Recognition:** Many systems lack the capability to recognize motion-based gestures, such as the ASL letter 'Z,' which require spatiotemporal analysis.
- **Environmental Robustness:** Variability in lighting, backgrounds, and camera angles still significantly impacts recognition accuracy.
- **End-to-End Solutions:** Few solutions address the full pipeline of sign language translation, from gesture recognition to natural language generation.

### 2.4. Commercial Solutions

Commercial systems, such as SignAll and Hand Talk, have attempted to bridge the communication gap using sign language recognition. While promising, these solutions typically require specialized hardware and controlled environments, making them inaccessible for broader adoption.

### 2.5. Our Contributions

Building upon prior work, our system integrates state-of-the-art techniques to overcome these limitations. By leveraging efficient preprocessing pipelines, robust CNN architectures, and real-time processing capabilities, we aim to deliver a practical, accessible solution for sign language translation on standard hardware.

## 3. Methodology

Our implementation combines efficient image processing with deep learning techniques to create a real-time sign language translation system. The system architecture emphasizes both accuracy and processing speed while maintaining accessibility through standard hardware requirements.

### 3.1. System Architecture

The system consists of three primary components working in a pipeline architecture:

1. **Input Processing:**
   - Reshaping input images
2. **Gesture Recognition Engine:**
   - Custom PyTorch CNN architecture
   - Multi-layer feature extraction pipeline
   - Batch normalization for training stability
   - GPU-accelerated inference using CUDA
3. **Output Generation System:**
   - Real-time text generation from recognized gestures
   - Confidence scoring for recognition accuracy

### 3.2. Data Collection and Preprocessing

#### 3.2.1 Dataset Composition

- **Primary Dataset:** Kaggle ASL Alphabet Dataset
  - 87,000 images for training
  - 26 classes (A-Z)
  - Diverse lighting conditions and backgrounds
- **Validation Data:**
  - 10% split from primary dataset
  - Custom recorded validation sequences
  - Real-world testing scenarios

#### 3.2.2 Preprocessing Pipeline

Our preprocessing workflow includes:

- **Image Preprocessing:**
  - Resolution standardization (224x224 pixels)
  - Normalization (mean=0.485, std=0.229)
  - Color space conversion (RGB to BGR)
  - Brightness and contrast adjustment

### 3.3. Model Architecture

Our CNN model architecture consists of:

- Input layer: 224x224x3
- 4 convolutional blocks with:
  - 3x3 convolutions
  - Batch normalization
  - ReLU activation
  - Max pooling
- 2 fully connected layers
- Softmax output layer (26 classes)

## 4. Implementation

Our implementation focuses on creating an efficient, real-time sign language translation system using Python, PyTorch, and OpenCV frameworks. The system is designed to run on standard hardware while maintaining robust performance.

### 4.1. Gesture Recognition Model

Our custom CNN architecture is implemented in PyTorch:

- **Model Architecture**
  - Input Layer: 224x224x3 RGB images
  - Convolutional Blocks:
    - Conv1: 64 filters, 3x3 kernel, ReLU
    - Conv2: 128 filters, 3x3 kernel, ReLU
    - Conv3: 256 filters, 3x3 kernel, ReLU
    - MaxPooling after each block
  - Fully Connected Layers:
    - FC1: 1024 units, ReLU
    - FC2: 512 units, ReLU
    - Output: 26 units (A-Z classes)
- **Training Configuration**
  - Loss Function: Cross-Entropy Loss
  - Optimizer: Adam (lr=0.001, 1=0.9, 2=0.999)
  - Batch Size: 32
  - Epochs: 50
- **Regularization Techniques**
  - Dropout (p=0.5) in FC layers
  - Batch Normalization
  - Early Stopping
  - L2 regularization (weight decay=1e-4)
- **Optimization Techniques**
  - GPU acceleration for landmark detection
  - Frame buffer management
  - Multi-threading for parallel processing

## 5. Experimental Results

This section presents a thorough evaluation of our sign language translation system, analyzing both quantitative metrics and qualitative user feedback.

### 5.1. Experimental Setup

#### 5.1.1 Hardware Configuration

Testing was conducted on the following setup:

- CPU: Intel i7-12700K
- GPU: NVIDIA RTX 3060 (12GB VRAM)
- RAM: 32GB DDR4
- Camera: Standard 1080p webcam (30 FPS)

#### 5.1.2 Testing Environment

Tests were conducted under various conditions:

- Indoor lighting (300-500 lux)
- Different backgrounds
- Various hand positions and angles
- Multiple users with different signing styles

### 5.2. Performance Metrics

#### 5.2.1 Model Performance

Table 1. Model Performance Metrics

| Metric | Value |
|---|---|
| Training Accuracy | 97% |
| Validation Accuracy | 98% |
| Average Latency | 40ms |
| Memory Usage | 2.5 GB |
| GPU Utilization | 60% |

#### 5.2.2 Letter Recognition Performance

Individual letter recognition accuracy varied across the alphabet:

- Overall Accuracy: 85%
- Best Performing Letters:
  - 'A': 95% accuracy
  - 'L': 94% accuracy
  - 'Y': 93% accuracy
- Challenging Letters:
  - 'E': 75% accuracy
  - 'M': 77% accuracy
  - 'N': 76% accuracy

### 5.3. System Performance Analysis

#### 5.3.1 Environmental Impact Analysis

Performance variation under different conditions:

Table 2. Environmental Impact on Accuracy

| Condition | Accuracy |
|---|---|
| Optimal Lighting | 85% |
| Low Light | 72% |
| Variable Distance | 78% |

### 5.4. Limitations and Challenges

Current limitations identified during testing:

- Decreased accuracy in low-light conditions
- Limited to single-hand gestures
- Unable to translate motion-based gestures
- Environmental dependency for optimal performance

The details of these limitations will be further elaborated in Section 6.

## 6. Discussion

Although our sign language has successfully achieved what we set out to achieve, it falls short due to several critical limitations. This section analyzes the system's current capabilities, limitations, and potential future directions.

### 6.1. System Capabilities

#### 6.1.1 Successful Features

Our implementation has demonstrated several key strengths:

- Real-time processing capability with
- High accuracy for basic ASL alphabet (85% overall)
- Minimal hardware requirements

Although these are desirable results, there are some critical limitations that we will address below.

### 6.2. Current Limitations

#### 6.2.1 Technical Constraints

1. Inability to translate motion-based gestures

   Currently, the model is unable to translate motion-based gestures, such as the letter 'Z'. Gestures like 'Z' require continuous tracking of hand motion to form the shape in the air. Since our model operates by checking each frame of signing if an alphabet is being signed, it is unable to recognize motion-based gestures, which have spatial and temporal context. This can be attributed to the absence of hand tracking, where having key-points and position of the hand is pivotal to translate these gestures.

2. Limited to single-hand gestures

   In sign languages, two-handed signs are commonly used for specific words, phrases, or concepts. Two-handed signs are quite common and can involve both hands performing the same or different movements, depending on the sign. This can also be attributed to the lack of hand tracking.

3. Vocabulary and context constraint

   Not only that, the model also faces a vocabulary constraint. As it stands, the model is limited to only ASL alphabet. Combined with former constraints, this meant that the model provides no support to translate common phrase, let alone sentences and conversations, which are sequences of signs with spatial and temporal contexts.

#### 6.2.2 Environmental Limitations

Since we are using pixels to predict and translate a given sign, we also run into environmental limitations.

1. Lighting sensitivity

   The model demonstrates significant sensitivity to variations in lighting conditions, with a reported 30%

drop in accuracy under poor lighting. Insufficient illumination often results in blurry or indistinct hand shapes, making it difficult for the model to correctly identify gestures. Additionally, the presence of backlighting—where the primary light source is behind the user—creates silhouette effects and shadows that further reduce the model's ability to discern hand signs accurately. Glare interference, caused by reflective surfaces or overly bright environments, introduces additional noise into the input data, compromising the model's recognition capabilities.

2. Background complexity

   The complexity of the background also plays a critical role in the model's performance. When operating in environments with busy or cluttered backgrounds, the model's accuracy decreases. This is largely due to the difficulty in isolating the hand pixels from the surrounding visual noise, which can confuse the gesture recognition process. Moving objects in the background exacerbate this problem by introducing dynamic changes that the model may mistakenly interpret as part of the gesture. Furthermore, the system's dependence on color contrast between the hands and the background can lead to inconsistencies in performance, particularly in scenarios where the background and hand colors are similar.

## 6.3. Future Improvements

### 6.3.1 Short-term Enhancements

To address the current limitations and enhance the system's performance, the following short-term implementation plan is proposed:

1. Integration of Mediapipe for Enhanced Gesture Tracking

   Previously, we decided to forgo the use of hand tracking and keypoint detection to improve on speed with minimal sacrifice on performance. However, the drawbacks are too severe to be ignored. As such, we will be integrating Mediapipe, an open-source framework by Google, which offers state-of-the-art hand tracking capabilities that can improve the robustness of gesture recognition. By leveraging Mediapipe's hand landmark detection, the system can achieve more accurate tracking of hand movements and positions, even in challenging lighting or background conditions. This integration will enable more precise input data for subsequent processing steps, ensuring better performance with minimal additional computational overhead.

2. Incorporation of MoViNet for Dynamic Gesture Recognition

   Building on top of enchanced gesture tracking, a transfer learning approach using MoViNet (Mobile Video Network) will be implemented. MoViNet is a lightweight, spatiotemporal model optimized for real-time video understanding. By leveraging MoViNet's ability to capture both spatial and temporal dependencies, the system can accurately process motion gestures, such as the ASL letter 'Z' and other dynamic signs. This accuracy is further enchanced with the extracted hand landmarks we get from using Mediapipe. Using pre-trained MoViNet models fine-tuned on a gesture-specific dataset, the system will transition from static frame analysis to efficient video-based gesture interpretation. This approach ensures low-latency recognition while maintaining high accuracy, making it suitable for real-time applications.

3. Utilization of the How2Sign Dataset

   The How2Sign dataset, a comprehensive collection of continuous sign language videos, will be utilized to train and fine-tune the system. This dataset includes a variety of static and dynamic gestures performed in diverse environments, offering a rich resource for training robust gesture recognition models. By incorporating How2Sign, the system will gain exposure to a wide range of signing styles, lighting conditions, and background complexities. Additionally, the dataset's multimodal nature—providing video, depth, and textual annotations—can be leveraged to enhance the model's understanding of spatiotemporal features. Further, synthetic augmentations, such as altering brightness, contrast, and motion speed, will expand the dataset to improve the system's generalizability in challenging scenarios.

### 6.3.2 Long-term Development Goals

To ensure the system's continued development and usability, several long-term improvements are envisioned.

1. Introduction of Webcam Support

   Introducing support for webcams is critical for system integration in the future, especially on mobile devices. Thus, a long-term goal of ours is to optimize the system such that it works efficiently with low-resolution and low-frame-rate webcams, which are commonly available in many consumer devices. Additionally, adaptive algorithms will be incorporated to dynamically adjust for varying frame rates and resolutions, ensuring consistent performance across different hardware. This enhancement will make the system more inclusive and practical for users with varying resources.

2. Cloud Deployment and API Integration

   Launching the gesture recognition system to the cloud and providing access through an API is a key long-term improvement that enhances scalability, accessibility, and versatility. By hosting the model on a cloud platform (e.g., AWS, Google Cloud, or Azure), all computationally intensive tasks, including gesture recognition and model inference, can be offloaded to the cloud, eliminating the need for high-performance local hardware. Users can interact with the system via a RESTful or gRPC API, enabling seamless integration into diverse applications such as mobile apps, web platforms, and enterprise tools.

   This approach ensures hardware independence, allowing any device with internet connectivity to access the system, while also supporting cross-platform compatibility. Cloud deployment enables centralized updates, ensuring users always have access to the latest model version without manual intervention. Additionally, cloud infrastructure supports horizontal scaling and load balancing to maintain consistent performance during peak usage.

3. Integration Possibilities The system's usability and impact can be amplified through integration with other technologies and applications:

   - **Assistive Technologies:** Integration with accessibility tools for the hearing-impaired, such as real-time captioning or translation systems, can enhance communication for individuals who rely on sign language.
   - **Virtual and Augmented Reality (VR/AR):** The system can be integrated into VR/AR platforms to create immersive learning environments for sign language education or virtual communication.
   - **Educational Applications:** Collaboration with e-learning platforms to develop interactive ASL tutorials can help broaden awareness and understanding of sign language.
   - **Smart Home Devices:** Incorporating gesture recognition into smart home ecosystems, such as controlling devices via ASL, will add another layer of functionality for users.

# 7. Conclusion

This paper presented a real-time sign language translation system utilizing computer vision and deep learning techniques. Our implementation achieves 85% accuracy in ASL alphabet recognition while maintaining real-time performance of 25 FPS on standard hardware. The system demonstrates the feasibility of automated sign language translation and its potential impact on breaking communication barriers.

## 7.1. Key Contributions

Our work makes several significant contributions to the field:

- Development of an efficient real-time processing pipeline
- Creation of an accessible, low-hardware-requirement solution
- Demonstration of practical sign language translation feasibility

## 7.2. Impact and Applications

The system shows promise for various real-world applications:

- Educational environments for deaf students
- Healthcare settings for patient communication
- Public service accessibility improvement
- Personal communication assistance

## 7.3. Final Remarks

While our current implementation has limitations, the results demonstrate the potential for computer vision-based sign language translation systems. The achieved performance metrics and user feedback suggest that such systems could significantly impact the deaf community's daily communication challenges. Future work will focus on expanding capabilities and improving accessibility, working toward a more inclusive communication solution for all users.

# References


[1] HowStuffWorks. How sign language works, 2024. Accessed: Nov. 11, 2024. 7

[2] Kaggle. Sign language mnist dataset. https://www.kaggle.com/datamunge/sign-language-mnist, 2024. Accessed: Dec. 11, 2024. 7

[3] World Federation of the Deaf. Sign language: Facts and figures, 2023. Accessed: Dec. 11, 2024. 1, 7

[4] Lionel Pigou, Sander Dieleman, Pieter-Jan Kindermans, and Benjamin Schrauwen. Sign language recognition using convolutional neural networks. In *European Conference on Computer Vision (ECCV) Workshops*, pages 572–578. Springer, 2014. 2, 7

[5] S. A. M. A. S. Senanayaka, R. A. D. B. S. Perera, W. Rankothge, S. S. Usgalhewa, H. D. Hettihewa, and P. K. W. Abeygunawardhana. Continuous American Sign Language Recognition Using Computer Vision and Deep Learning Technologies. In *2022 IEEE Region 10 Symposium (TENSYMP)*, pages 1–6, 2022. 2, 7

[6] Thad Starner, Joshua Weaver, and Alex Pentland. Real-time american sign language recognition using desk and wearable computer based video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(12):1371–1375, 1998. 1, 7

[1] [2] [3] [4] [5] [6]